Webapplicaties I

Hoofdstuk 8 - Grid Layout

HO GENT

Inleiding

- Deze PowerPoint is opgedeeld in vier delen die elk gebaseerd zijn op een MDN-artikel:
  - **1. Basisconcepten grid layout**
    Gebaseerd op: Basic Concepts of grid layout
  - **2. Items positioneren op de grid**
    Gebaseerd op: Line-based placement with CSS Grid
  - **3. Grid-template-areas property**
    Gebaseerd op: grid-template-areas
  - **4. Box alignment in CSS Grid Layout**
    Gebaseerd op: Box alignment in CSS Grid Layout
- De volledige officiële grid layout specification vind je op: https://www.w3.org/TR/css-grid-1/

HO GENT

1. Basisconcepten Grid Layout

Referenties:

- MDN Basic Concepts of grid layout (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout)
- De volledige officiële grid layout specification: https://www.w3.org/TR/css-grid-1/

HO GENT

Flexible Box Layout vs Grid Layout

- In tegenstelling tot ‘Flexible Box layout’ (kort Flexbox of flex), dat een eendimensioneel lay-outsysteem is, is ‘Grid Layout’ (kort Grid) een tweedimensioneel lay-outsysteem. Zie afbeelding volgende dia.
- Grid en flex kunnen samen gebruikt worden om complexe lay-outs te maken.

Flexible Box Layout vs Grid Layout

Figure 1 *Exemplary Flex Layout Example*

Figure 2 *Exemplary Grid Layout Example*

Bron: https://www.w3.org/TR/css-grid-1/#intro, laatst geraadpleegd op 2019-08-12

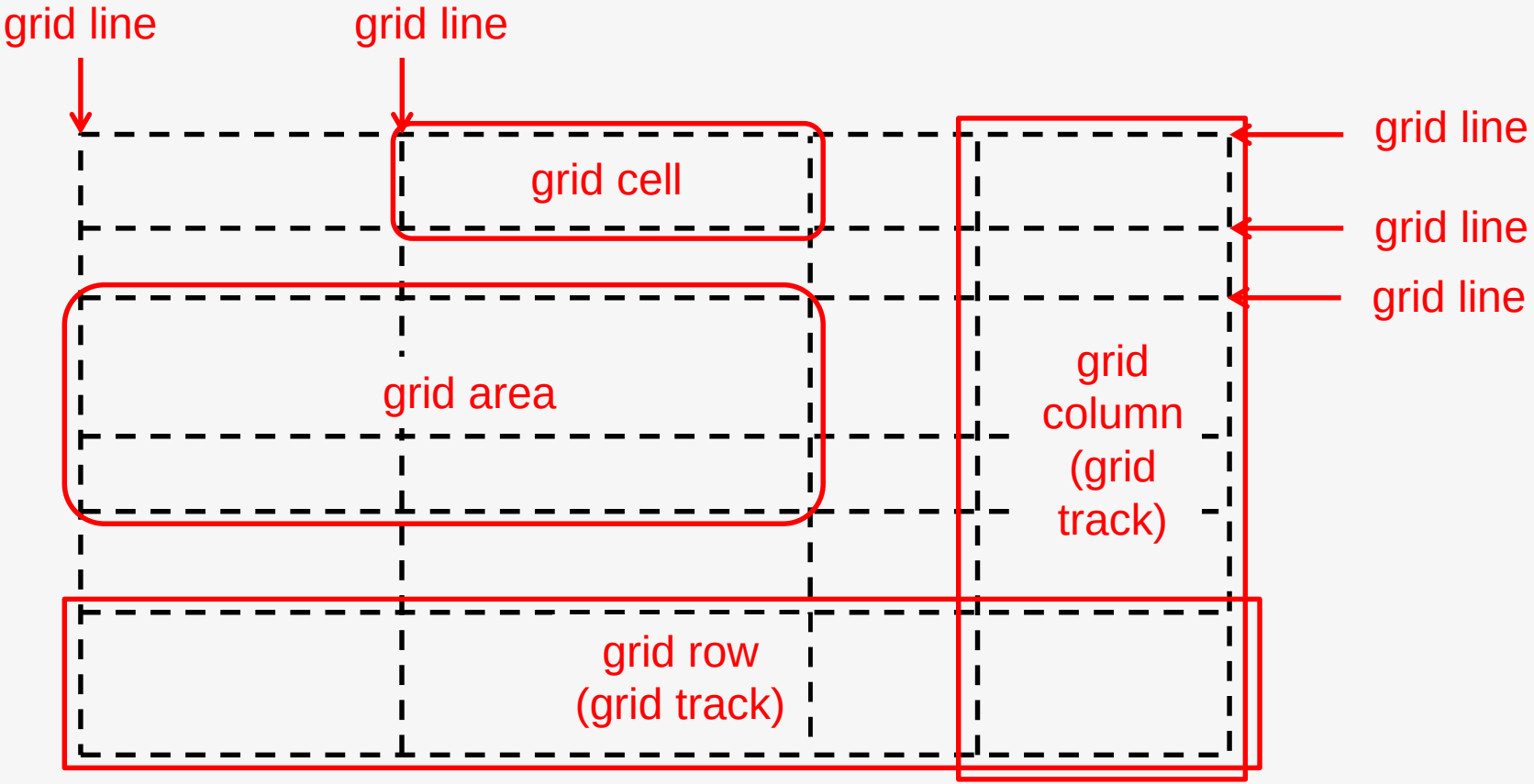
HO GENT

Wat is een grid?

- Een *grid* (of raster) bestaat uit (denkbeeldige) horizontale en verticale lijnen die een *grid container* opdelen in rijen, kolommen en cellen. Te vergelijken met een tabel.
- Op de volgende dia vind je een afbeelding van een grid waarop enkele grid termen zijn aangeduid.
  Meer info vind je op
  https://www.w3.org/TR/css-grid-1/#grid-concepts

HO GENT

Grid terminology



Waarom CSS grid layout?

- Zorgt voor een goede scheiding tussen inhoud (html) en opmaak (css). We kunnen een html-element een andere positie op de grid geven zonder dat we daarvoor de markup (html) moeten wijzigen.

HO GENT

Algemene procedure gebruik Grid Layout

1. Creëer een *grid container*-element. Hiermee definieer je impliciet ook de *grid items*.
2. Definieer de *grid* en plaats de *grid items* op de grid.

HO GENT

De grid container en de grid

- Je creëert een *grid container*-element met `display: grid` of `display: inline-grid`. Waarbij `display: grid` een block-level-element creëert en `display: inline-grid` een inline-level-element. Het creëren van een *grid container* heeft tevens tot gevolg dat alle (directe) kind-elementen van het *grid container*-element, *grid items* worden.
- De *grid* zelf, het aantal rijen en/of kolommen en hun hoogte/breedte, definieer je via de `grid-template-rows` en `grid-template-columns` properties (alsook via de property `grid-template-areas` zie later)

HO GENT

Voorbeeld: grid container

- In het volgende voorbeeld definiëren we een grid container met een grid bestaande uit drie kolommen van elk **200px** breed.
- Aangezien we niet expliciet het aantal en de hoogte van de rijen opgeven, zullen er automatisch rijen worden aangemaakt, waarbij ook de hoogte automatisch bepaald wordt.
- Er is in dit voorbeeld geen expliciete plaatsing van de items, waardoor de grid items in ‘HTML source order’ in de grid geplaatst zullen worden. Hoe je grid items expliciet op een bepaalde plaats in de grid kunt plaatsen komt later aan bod.

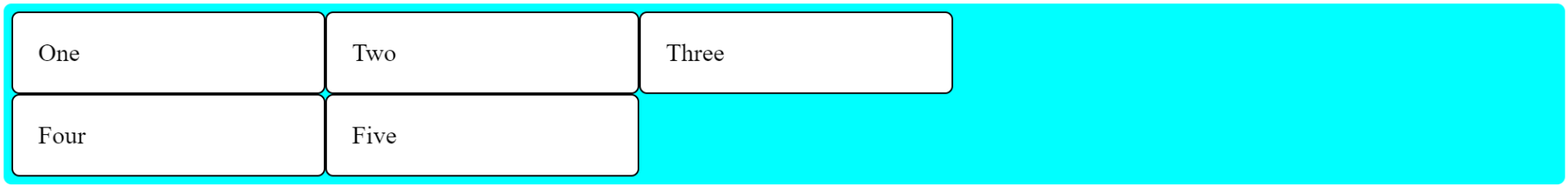
HO GENT

HTML

```html
<div class="grid-container">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

```css
.grid-container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```



Grid 01: grid container (codepen.io)

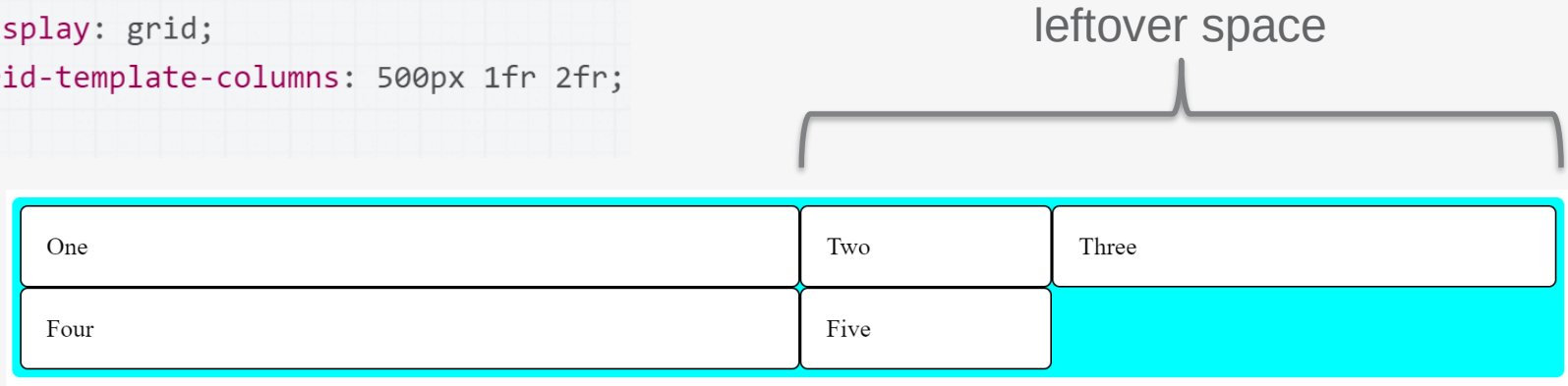
De fr-eenheid

- Met de *fr-eenheid* kan je ‘flexible grid tracks’ (rijen of kolommen) maken.
- De *fr-eenheid* stelt een fractie voor van de ‘leftover space’. De ‘leftover space’ is de ruimte die niet ingenomen wordt door ‘non-flexible grid tracks’.

HO GENT

```
/* Voorbeeld met één non-flexible column track (500px) en twee
flexible column tracks.
De ‘leftover space’ wordt verdeeld over de tweede en de derde
kolom. Hiervoor wordt de ‘leftover space’ eerst in drie gedeeld
en hiervan krijgt de tweede kolom één deel en de derde kolom
twee delen.*/
```

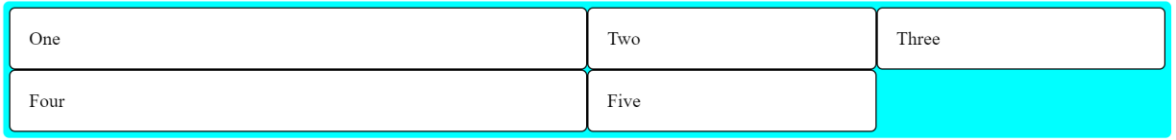
```
.grid-container {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```



[Grid 02: fr-eenheid (codepen.io)]

```
/* Voorbeeld met één kolom met breedte van 2fr en twee kolommen met een breedte van 1fr. De beschikbare ruimte wordt dus in vier gedeeld. De eerste kolom krijgt twee delen en de twee overige kolommen elk één deel.*/

.grid-container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```




```
/* Voorbeeld met drie kolommen van gelijke breedte, die automatisch groeien of krimpen afhankelijk van de beschikbare ruimte*/

.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

HO GENT

repeat()-functie

Met de `repeat`()-functie kan je (een deel van) een ‘track list’ herhalen.

- Zo kan je
  ```
  grid-template-columns: 1fr 1fr 1fr;
  ```
  korter schrijven als:
  ```
  grid-template-columns: repeat(3, 1fr);
  ```
- Een meer uitgebreid voorbeeld:
  ```
  grid-template-columns: 20px repeat(4, 1fr 2fr);
  ```
  is hetzelfde als:
  ```
  grid-template-columns: 20px 1fr 2fr 1fr 2fr 1fr 2fr 1fr 2fr;
  ```


HO GENT

De ‘implicit grid’ en de ‘explicit grid’

- In de vorige voorbeelden hebben we steeds de kolommen (column tracks) expliciet gedefinieerd met de `grid-template-columns`-property en de rijen werden automatisch gecreëerd. Deze rijen maken deel uit van de ‘implicit grid’.
- Rijen en kolommen gedefinieerd in de ‘implicit grid’ zijn standaard ‘auto-sized’, maar je kan hiervoor een hoogte/breedte instellen via de properties `grid-auto-rows` en `grid-auto-columns`.

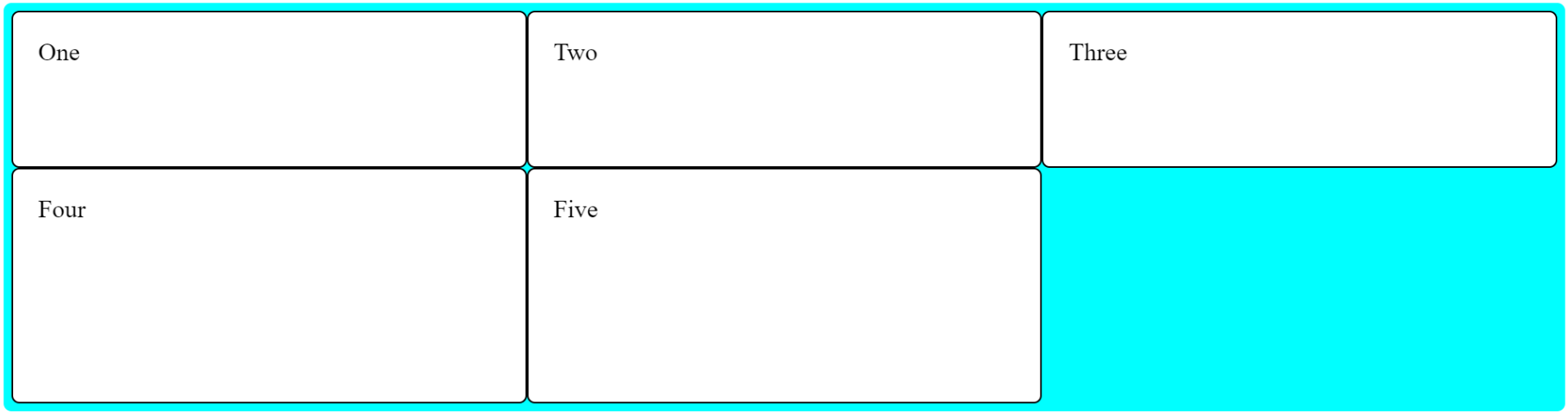
HO GENT

HTML

```html
<div class="grid-container">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

```css
/* de hoogte van de eerste rij stellen we in op 100px en eventuele bijkomende impliciet gecreëerde rijen krijgen een hoogte van 150px*/
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px;
  grid-auto-rows: 150px;
}
```



[Grid 03: implicit grid (codepen.io)]

minmax()-functie

- Voorbeeld: de standaard `auto` voor de rijhoogte betekent dat de hoogte van de rij aangepast zal worden aan de inhoud. Maar als we willen dat de rijhoogte nooit kleiner wordt dan `100px`, dan kunnen we gebruikmaken van de `minmax()`-functie bij het definiëren van de ‘row tracks’.

```
grid-auto-rows: minmax(100px, auto);
```

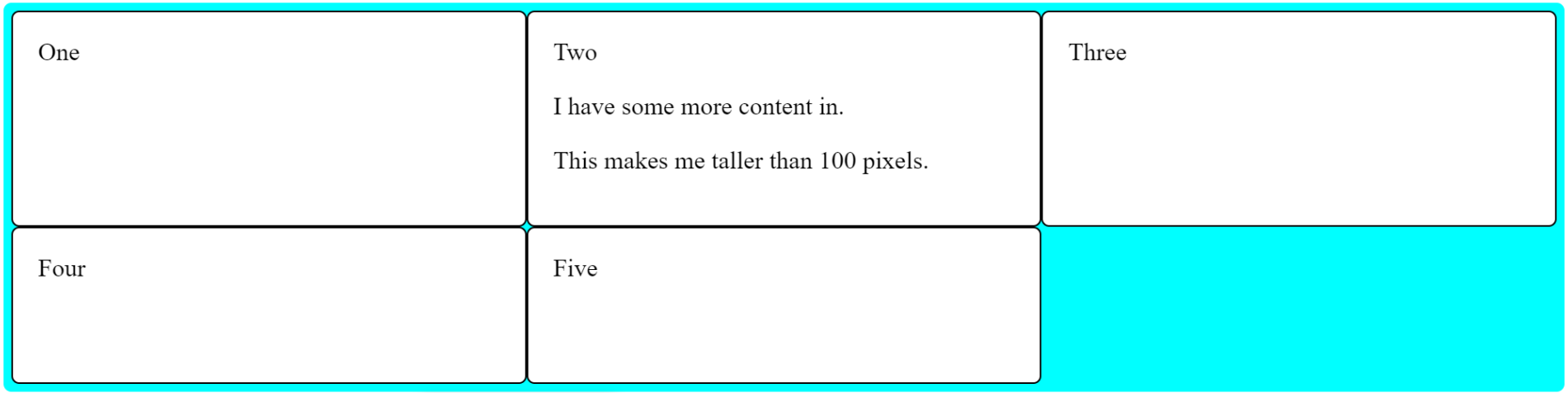

HO GENT

HTML

```html
<div class="grid-container">
 <div>One</div>
  <div>Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

```css
/* de rijhoogte van de rijen is minimaal 100px en maximaal de hoogte van de inhoud.*/
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);
}
```



Grid 04: minmax()-notatie (codepen.io)

Repeat-to-fill: auto-fill en auto-fit

- In plaats van bij de `repeat`()-functie, als eerste argument een getal op te geven, kan je ook `auto-fill` of `auto-fit` gebruiken.
- Voorbeeld (zie volgende dia):

```
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```

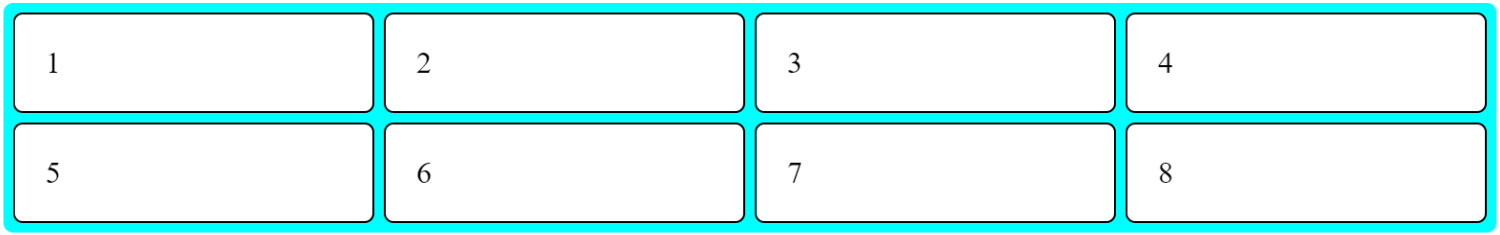
We geven zelf geen vast aantal kolommen op bij de `repeat`()-functie maar laten de browser het aantal kolommen bepalen (`auto-fill`).
Bijvoorbeeld bij 800px is er niet genoeg ruimte voor 6 kolommen van 150px (6*150px=900px) en een kolom mag niet smaller zijn dan 150px, dus maakt de browser 5 kolommen.

HO GENT

Voorbeelden: repeat()-functie

```
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```


```
grid-template-columns: repeat(4, 1fr);
```



Grid 04bis: repeat()-functie (codepen.io)

Meer info over auto-fill en auto-fit

- `Auto-fill` en `auto-fit` werken op dezelfde manier, behalve als er lege tracks zijn.
- Een goede video over hoe `auto-fill` en `auto-fit` werken en wat het verschil is tussen de twee vind je op de website van https://gridbyexample.com van Rachel Andrew.

  Videolink:

  https://gridbyexample.com/video/series-auto-fill-auto-fit/

HO GENT

2. Items positioneren op de grid

Referenties:

- MDN Line-based placement with CSS Grid (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Line-based_Placement_with_CSS_Grid)
- De volledige officiële grid layout specification: https://www.w3.org/TR/css-grid-1/

HO GENT

Grid Lines

- Als je een grid definieert dan krijgen de lijnen waaruit de grid bestaat automatisch nummers. Je kan deze lijnnummers gebruiken om een ‘grid item’ expliciet te positioneren in de grid.
- Op de volgende twee dia’s wordt telkens dezelfde webpagina weergeven in de browser Mozilla Firefox, maar in de tweede schermafbeelding zijn via de *Grid Inspector* in de *Firefox Developer Tools* de *Grid lines* en hun *Line numbers* zichtbaar gemaakt.

HO GENT

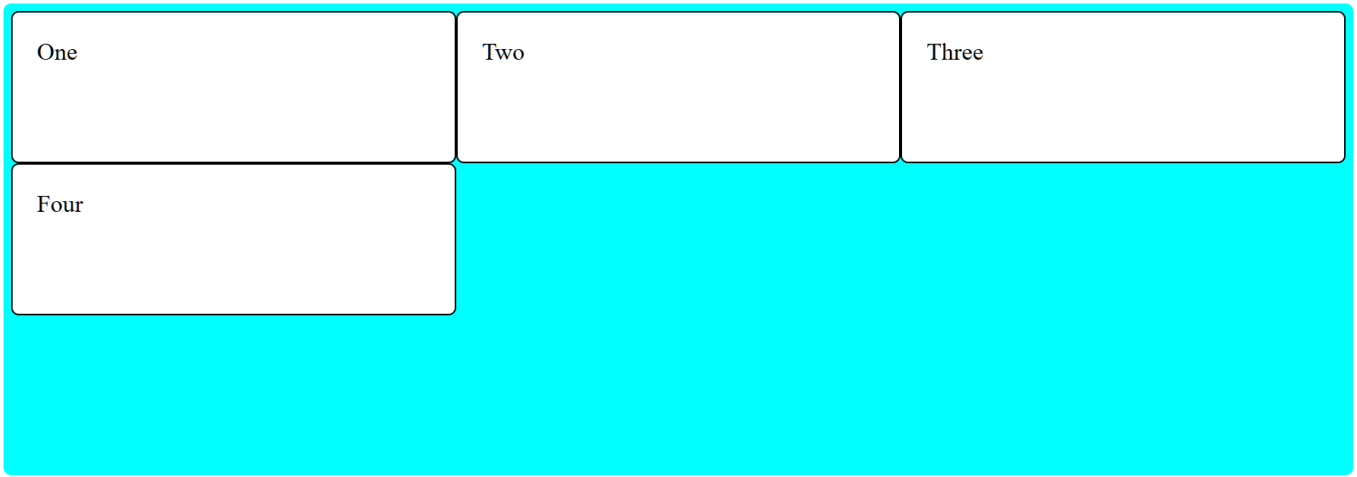
Afbeelding in Firefox

HTML

```
<div class="grid-container">
    <div class="box1">One</div>
    <div class="box2">Two</div>
    <div class="box3">Three</div>
    <div class="box4">Four</div>
</div>
```

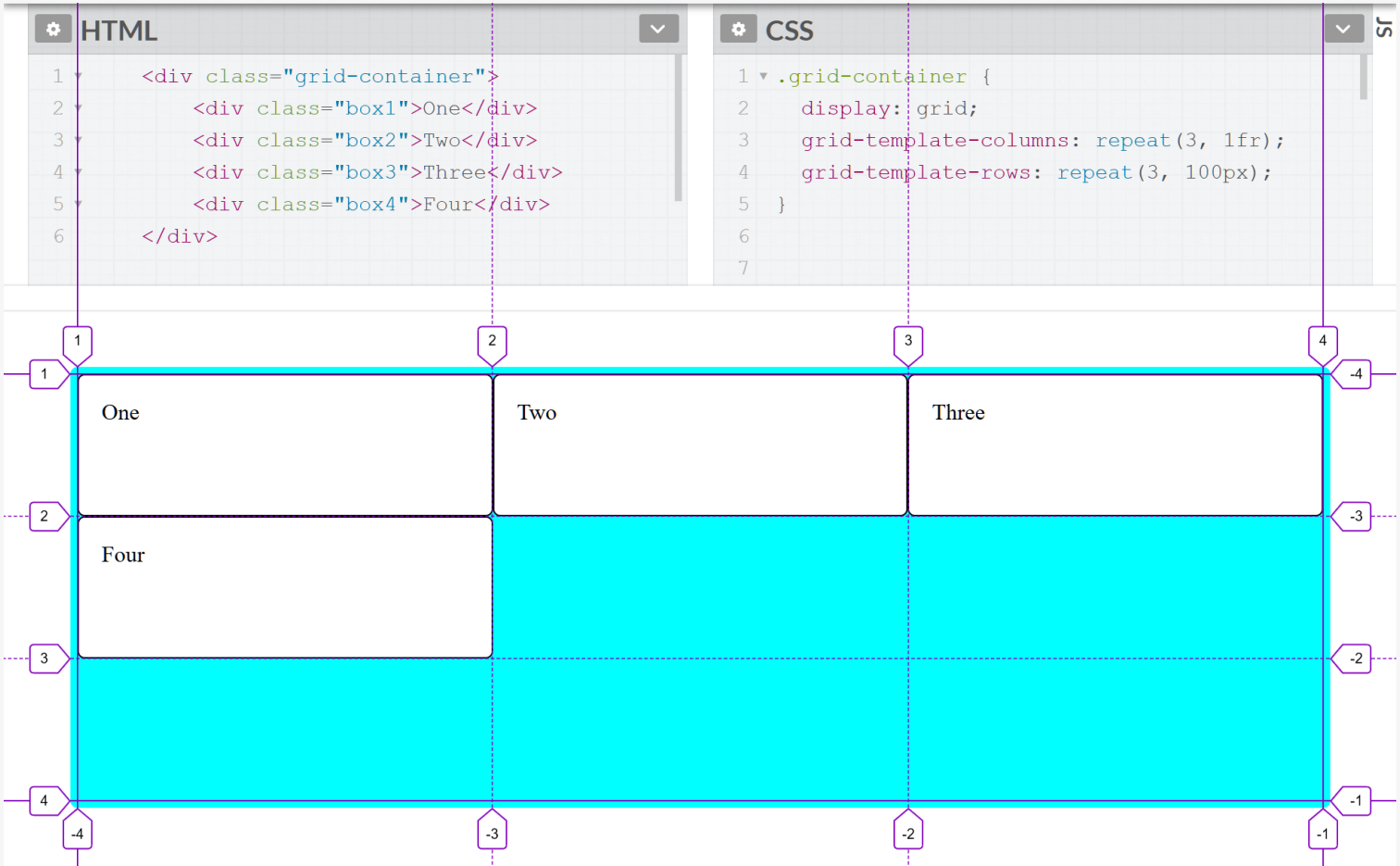
CSS

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 100px);
}
```



[Grid 05: grid lines (codepen.io)]

Afbeelding met Grid Lines en Line Numbers zichtbaar via de Firefox Grid Inspector



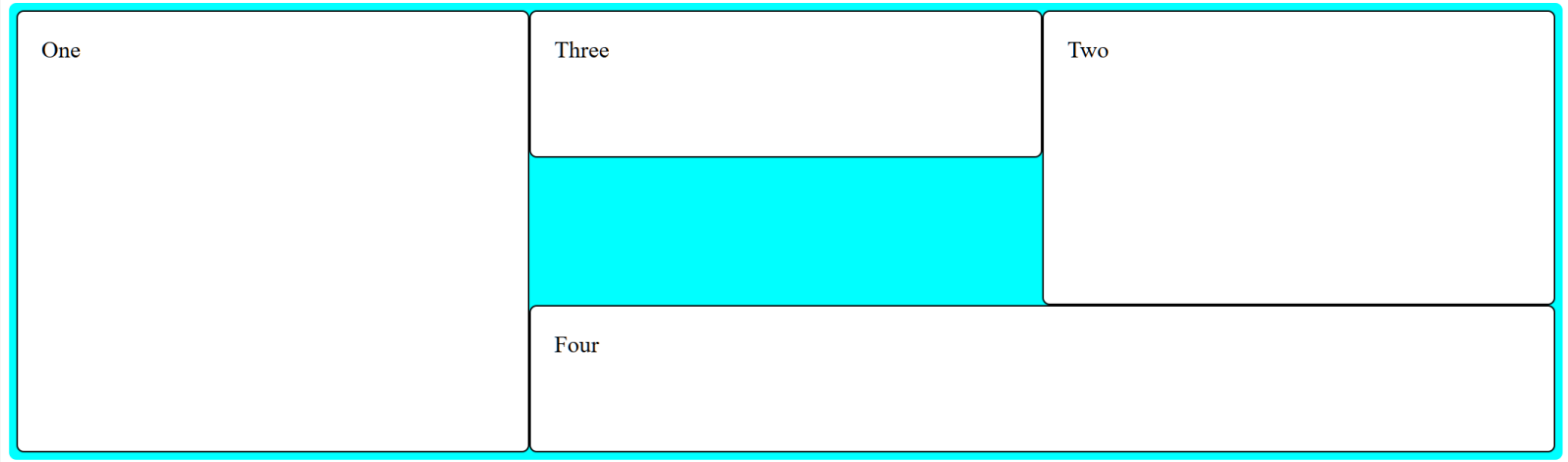
Grid Lines

- De Grid Inspector in Mozilla Firefox is een goede tool om te leren werken met grid.
- Merk op dat de getallen verwijzen naar de lijnen en niet naar de kolommen of de rijen. Zo bevat een 3-column-layout 4 lijnen.

HO GENT

Grid Items positioneren op de Grid met Line Numbers

```
.box1 {grid-column-start: 1;grid-column-end: 2;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-column-end: 4;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-column-end: 3;grid-row-start: 1;grid-row-end: 2;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;grid-row-end: 4;}
```



[Grid 06: position grid Items with grid lines (codepen.io)]

Default Spans

- In het vorige voorbeeld werd elke ‘grid-row-end’ en ‘grid-column-end’ expliciet vermeld. Als het item echter maar één track (rij of kolom) overspant, mag je de ‘grid-row-end’ of ‘grid-column-end’ waarden weglaten. Het vorige voorbeeld kan dus ook geschreven worden als:

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```


HO GENT

Gutters

- Witruimte tussen kolommen en rijen, genoemd ‘Gutters’, creëer je met `column-gap` en `row-gap` of de shorthand `gap`.
- **Opmerking** Sommige browsers gebruiken nog de oude notaties met de prefix `grid-` nl. `grid-column-gap`, `grid-row-gap` en `grid-gap`, maar de meeste browser-makers hebben hun browser reeds aangepast.

HO GENT

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 50px 50px;
  column-gap: 10px;
  row-gap: 1rem;
}
```

Grid 07: gutters (codepen.io)

Het sleutelwoord ‘span’

- In plaats van het ‘end’ lijnnummer op te geven, kan je ook het aantal tracks opgeven dat je wilt overspannen.
- Voorbeeld:

```
grid-row-start: 1;grid-row-end: 4;}
```

kan je dus ook schrijven als

```
grid-row-start: 1;grid-row-end: span 3;}
```

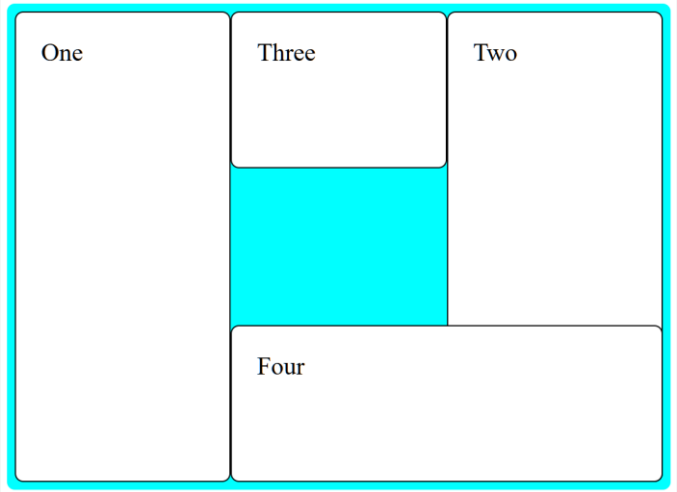

HO GENT

Overlappende grid items

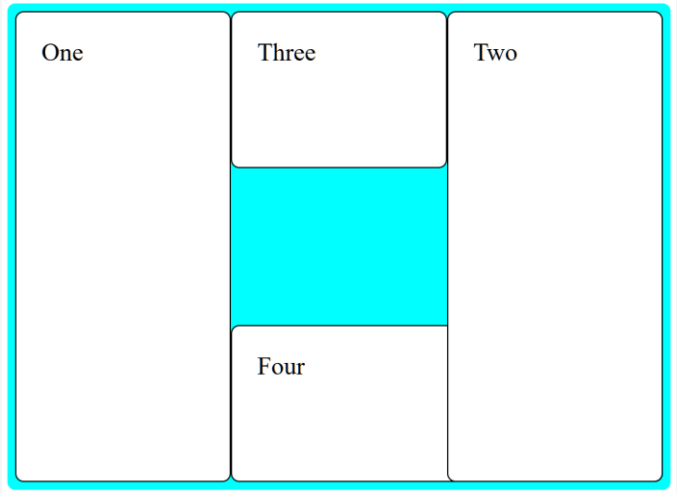
- Grid items kunnen overlappen.
- Als we in het voorbeeld op de volgende dia `.box2` laten eindigen bij rijlijn 4 i.p.v. bij rijlijn 3, dan zit `.box2` gedeeltelijk verscholen achter `.box4`. Wensen we `.box2` naar de voorgrond te brengen dan moeten we de `z-index`-property aanpassen, zoals we gezien hebben bij *positioned layout*.

HO GENT

```css
.box2 {
  grid-column-start: 3;
  grid-row-start: 1;
  grid-row-end: 4;
}
```




```css
.box2 {
  grid-column-start: 3;
  grid-row-start: 1;
  grid-row-end: 4;
  z-index: 1;
}
```



Grid items plaatsen

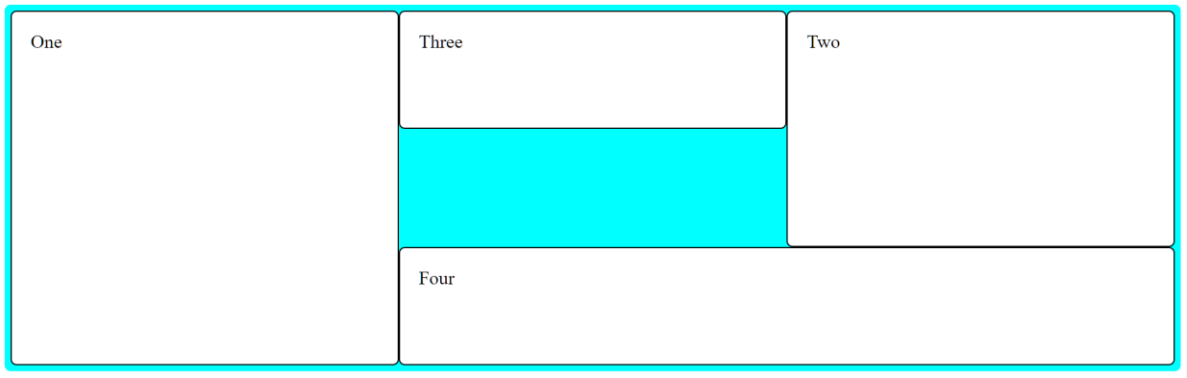
- Om *grid items* in de *grid* te plaatsen bestaan er naast de grid-placement properties `grid-column-start`, `grid-column-end`, `grid-row-start` en `grid-row-end` ook de shorthands `grid-column`, `grid-row` en `grid-area`.

grid-area			
grid-column		grid-row	
grid-column-start	grid-column-end	grid-row-start	grid-row-end

Bron: https://www.w3.org/TR/css-grid-1/#common-uses

- Op de volgende dia's vind je, hoe je Voorbeeld 06 korter kan schrijven met deze properties. Meer info over het plaatsen van grid items vind je o.a. in de specification: https://www.w3.org/TR/css-grid-1/#placement

HO GENT



Voorbeeld 06…

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```

…kan korter geschreven worden met de `grid-column` en `grid-row` shorthand properties

```
.box1 {grid-column: 1;grid-row: 1 / 4;}
.box2 {grid-column: 3;grid-row: 1 / 3;}
.box3 {grid-column: 2;grid-row: 1;}
.box4 {grid-column: 2 / 4;grid-row: 3;}
```

Achterwaarts tellen

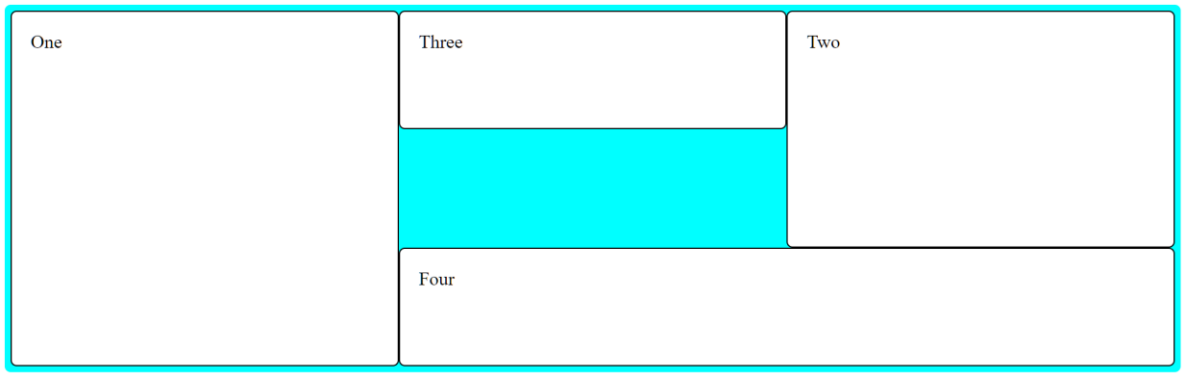
- Het is ook mogelijk om achterwaarts te tellen bij het opgeven van de lijnnummers. Zo kan je in Voorbeeld 06 het lijnnummer 4 vervangen daar lijnnummer -1 en kan

```
.box1 {grid-column:1; grid-row: 1 / 4;}
```

korter geschreven worden als

```
.box1 {grid-column:1; grid-row: 1 / -1;}
```


HO GENT



Voorbeeld 06…

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```

…kan ook korter geschreven worden met de `grid-area` property

```
.box1 {grid-area: 1 / 1 / 4 / 2;}
.box2 {grid-area: 1 / 3 / 3 / 4;}
.box3 {grid-area: 1 / 2 / 2 / 3;}
.box4 {grid-area: 3 / 2 / 4 / 4;}
```

```
/* De volgorde van de waarden
voor de grid-area property is:
  grid-row-start
  grid-column-start
  grid-row-end
  grid-column-end */
```

3. Grid-template-areas property

Referenties:

- MDN grid-template-areas property (https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-areas)
- De volledige officiële grid layout specification: https://www.w3.org/TR/css-grid-1/

HO GENT

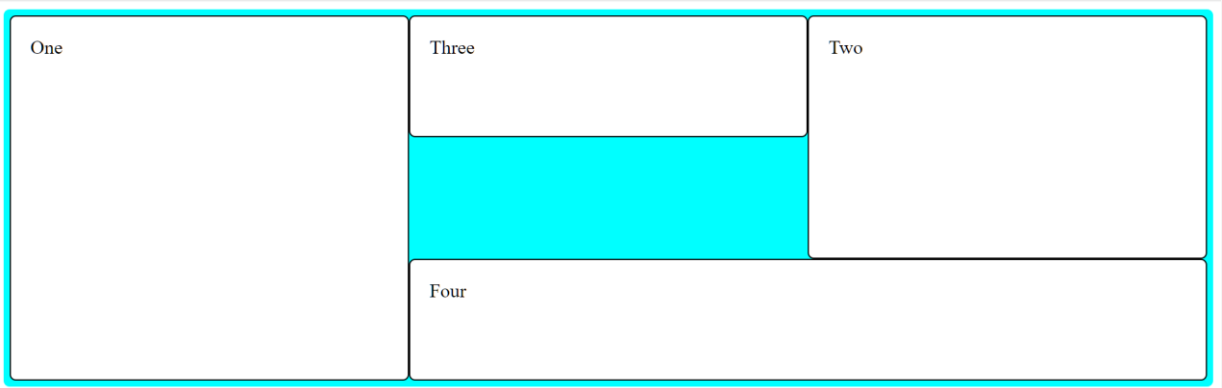
grid-template-areas property

- Een andere manier om grid items in de grid te plaatsen is met de `grid-template-areas` property. Hiermee creëer je namen voor grid areas die je dan kan gebruiken in bijv. de `grid-area` property om grid items te plaatsen in een grid area.
- Op de volgende dia vind je Voorbeeld 06 herschreven met de `grid-template-areas` property. Merk op dat:
  - je meerdere rijen of kolommen kan overspannen door de naam te herhalen;
  - je lege grid cellen maakt met een punt: .
  - de waarde van `grid-template-areas` de structuur van de grid visualiseert.

HO GENT

Voorbeeld 06 herschreven met de `grid-template-areas` property

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 100px);
  grid-template-areas:
  "one three two"
  "one .     two"
  "one four  four";
}
```




```
.box1 {grid-area: one;}
.box2 {grid-area: two;}
.box3 {grid-area: three;}
.box4 {grid-area: four;}
```

Grid 06bis: grid-template-areas (codepen.io)

Opmerking ‘Named Grid Lines’

- Je kan niet alleen namen geven aan grid areas, maar ook aan grid lines.
- Wellicht zal je echter meestal werken met ‘Named areas’ (`grid-template-areas` property), bijgevolg zullen we het werken met 'Named Grid Lines' niet bespreken. Wie hierin toch geïnteresseerd is zie: MDN - Layout using named grid lines

HO GENT

4. Box alignment in CSS Grid Layout

Referenties:

- MDN Box alignment in CSS Grid Layout (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Box_Alignment_in_CSS_Grid_Layout)
- De volledige officiële CSS Box Alignment specification (W3C Working Draft) https://www.w3.org/TR/css-align-3/

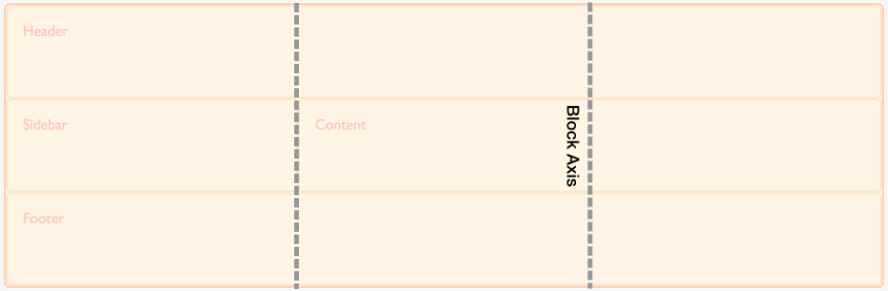
HO GENT

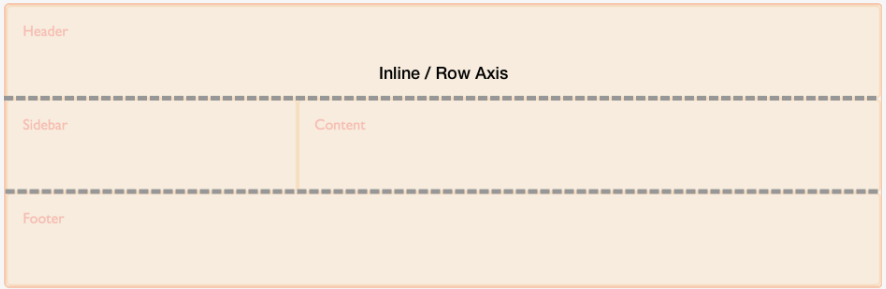
Box alignment in CSS Grid Layout

- Er zijn veel gelijkenissen tussen de werking van Box alignment in flex en in grid, maar er zijn ook verschillen.

HO GENT

Box alignment in CSS Grid Layout





Om items uit te lijnen langs de *block axis* gebruik je de properties align-items en align-self. Items uitlijnen langs de *inline axis* doe je met justify-items and justify-self.

HO GENT

We werken met de voorbeelden van de MDN-website. Deze voorbeelden gebruiken allemaal de volgende basis HTML en CSS …

HTML

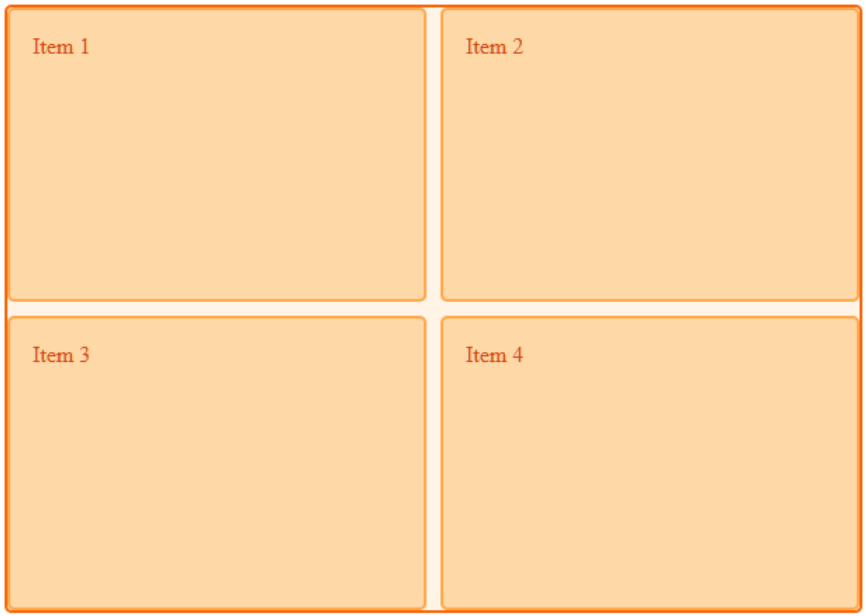
```
<div class="wrapper">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div>
```

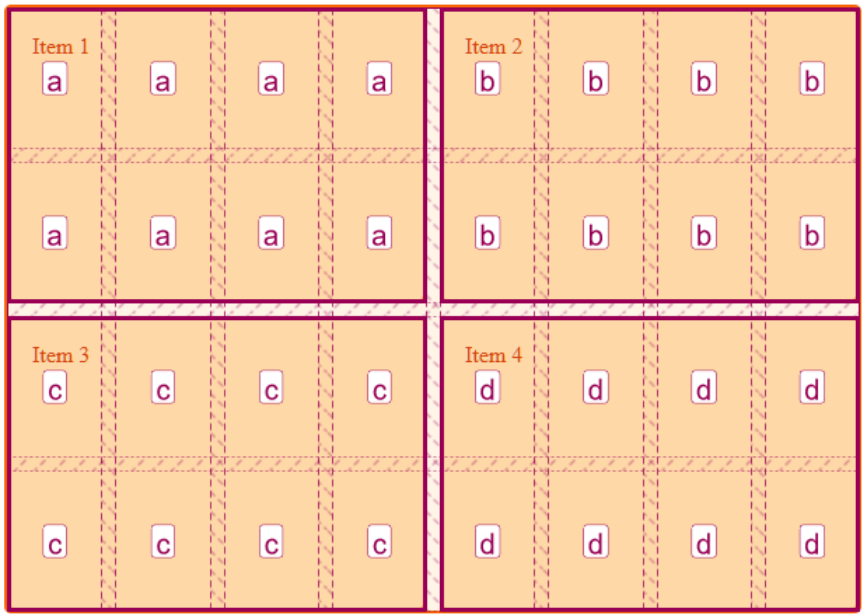
CSS

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(8, 1fr);
  grid-gap: 10px;
  grid-auto-rows: 100px;
  grid-template-areas:
    "a a a a b b b b"
    "a a a a b b b b"
    "c c c c d d d d"
    "c c c c d d d d";
}

.item1 {grid-area: a;}
.item2 {grid-area: b;}
.item3 {grid-area: c;}
.item4 {grid-area: d;}
```

… met het onderstaande als resultaat. In de rechtse afbeelding is de Firefox Grid Inspector gebruikt om de grid te visualiseren.
In dit startvoorbeeld zijn er geen expliciete Box alignment properties ingesteld;





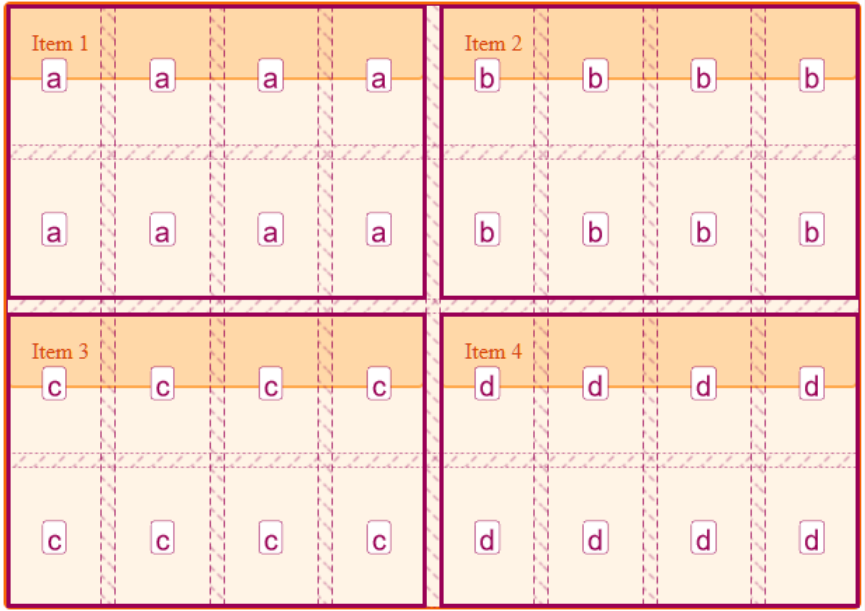
Block axis: align-self en align-items

- De `align-self` property stelt de uitlijning in van een grid item binnen zijn grid area/cell langs de block axis.
- De default voor `align-self` komt overeen met `stretch`.
  Bij elke andere waarde worden de afmetingen van het item automatisch berekend zodat de inhoud er net in past.
- De `align-items` property stelt de `align-self` property in voor alle grid-items (directe kinderen van de grid container).
  De `align-items` property stel je in op de grid container.

HO GENT

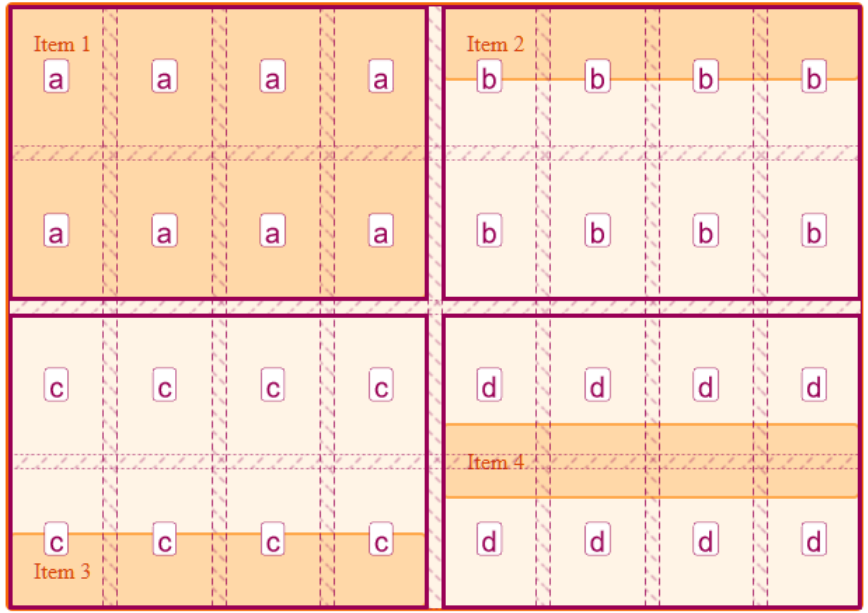
Voorbeeld 1a: items uitlijnen langs de block axis met `align-self` en `align-items`

```
.wrapper {
  align-items: start;
}
```



Voorbeeld 1b: items uitlijnen langs de block axis met `align-self` en `align-items`

```
.item2 {align-self: start;}
.item3 {align-self: end;}
.item4 {align-self: center;}
```



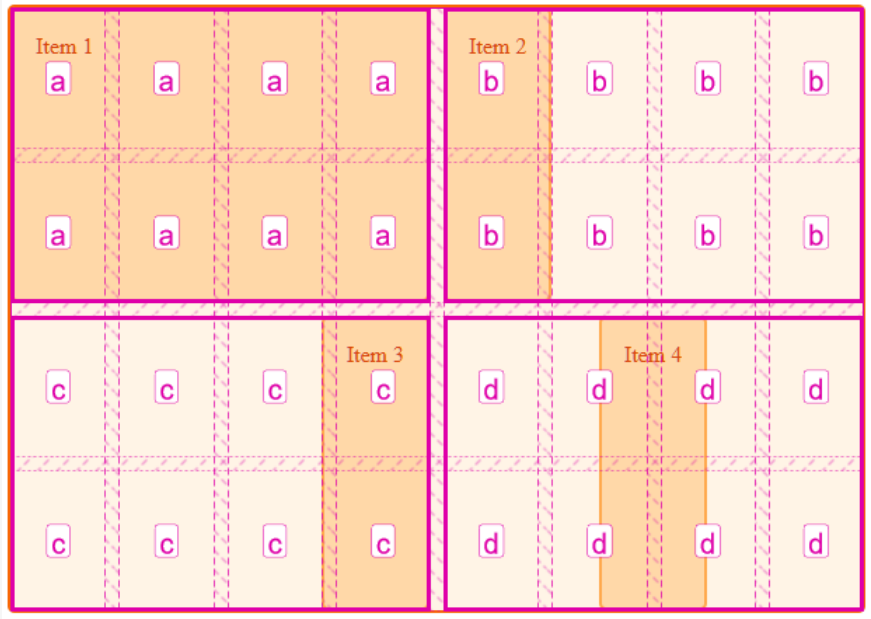
Inline axis: justify-self en justify-items

- De `justify-self` property stelt de uitlijning in van een grid item binnen zijn grid area/cell langs de inline axis.
- De default voor `justify-self` komt overeen met `stretch`.
  Bij elke andere waarde worden de afmetingen van het item automatisch berekend zodat de inhoud er net in past.
- De `justify-items` property stelt de `justify-self` property in voor alle grid items (directe kinderen van de grid container).
  De `justify-items` property stel je in op de grid container.

HO GENT

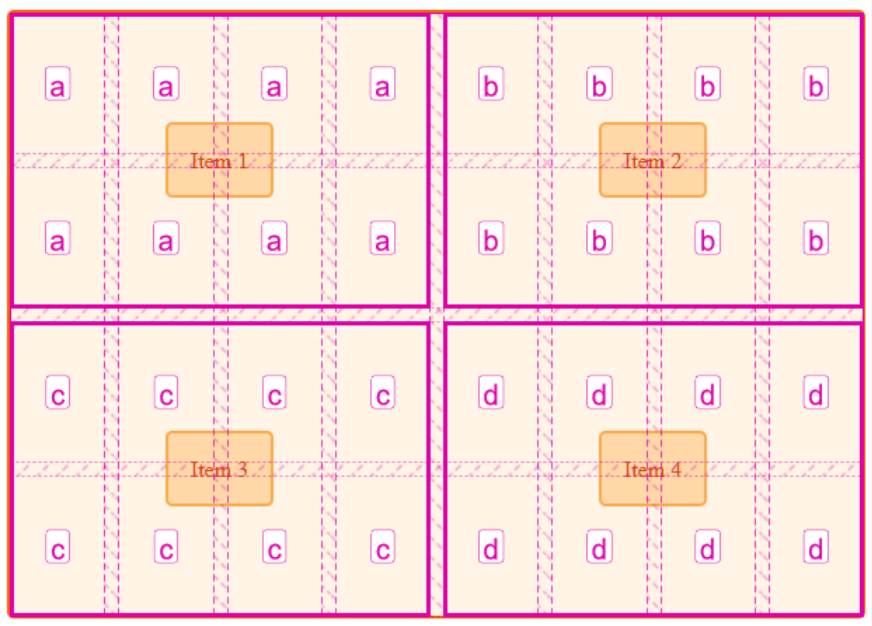
Voorbeeld 2: items uitlijnen langs de inline axis met `justify-self` en `justify-items`

```
.item2 {justify-self: start;}
.item3 {justify-self: end;}
.item4 {justify-self: center;}
```



Voorbeeld 3: items centreren
met `align-self/align-items` en `justify-self/justify-items`

```
.wrapper {
  align-items: center;
  justify-items: center;
}
```



align-content en justify-content

- Deze properties worden gebruikt als de grid tracks (rijen of kolommen) niet de volledige grid container innemen. Je kan in dit geval de tracks zelf uitlijnen binnen de container.
- Voor meer info en voorbeelden zie terug:
  MDN Box alignment in CSS Grid Layout (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Box_Alignment_in_CSS_Grid_Layout)

HO GENT

Bijlage

- Klik hier voor een overzicht van de gebruikte ‘CodePens’.

HO GENT

Extra info (geen examenstof)

- In deze PowerPoint wordt niet alles besproken van Grid Layout. Nog een interessant artikel is bijvoorbeeld:
  MDN - Auto-placement in CSS Grid Layout

HO GENT